

Traitement du signal et Apprentissage profond ; applications industrielles

Thomas COURTAT
thomas.courtat@thalesgroup.com

Master MVA 2023



- 1 Apprentissage profond 2
 - Choix de modèle
 - Espace latent
 - Séparation de sources

Apprentissage profond 2

Choix de modèle

Recherche d'hyperparamètres

Hyperparamètres \mathcal{H} =

- Paramètres d'architecture: nombre de couches, taille des noyaux, dimension état caché ...
- Paramètres d'optimisation: type d'optimiseur, learning rate, taille batch, régularisation...

Apprentissage pour $H \in \mathcal{H} : H \rightarrow V(H)$ score en validation de H .

- Recherche sur grille
- Recherche par tirage aléatoire
- Recherche dirigée selon un algorithme de décision

L'évaluation finale se fait sur un ensemble de test indépendant de l'ensemble de validation.

Deux bibliothèques interfacées en Python: HyperOpt et Ray[Tune]

HyperBand

Algorithme de recherche d'hyperparamètres pour algorithmes dont l'entraînement est itératif.

On se donne:

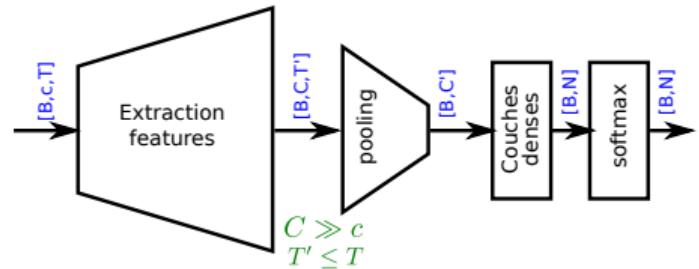
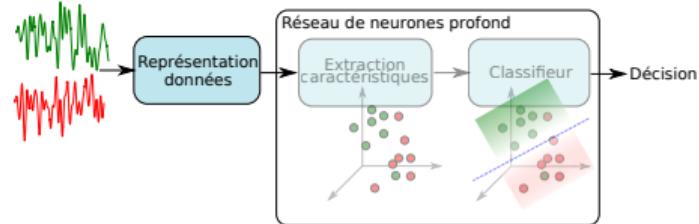
- Une fonction d'initialisation: $H \rightarrow A$ algorithme
- Pour A une méthode `.train(n_iter)`
- Pour A une méthode `.evaluate(dataset)`
- Un facteur de décimation Δ et un nombre d'itérations maximal M
- On tire Δ^M configurations H_i
- Pour chaque H_i on initialise un algorithme A_i
- On entraîne chaque A_i sur 1 itération et on évalue chaque modèle
- On garde les Δ^{M-1} meilleurs algorithmes
- On poursuit leur entraînement sur Δ itérations
- ...

Espace latent

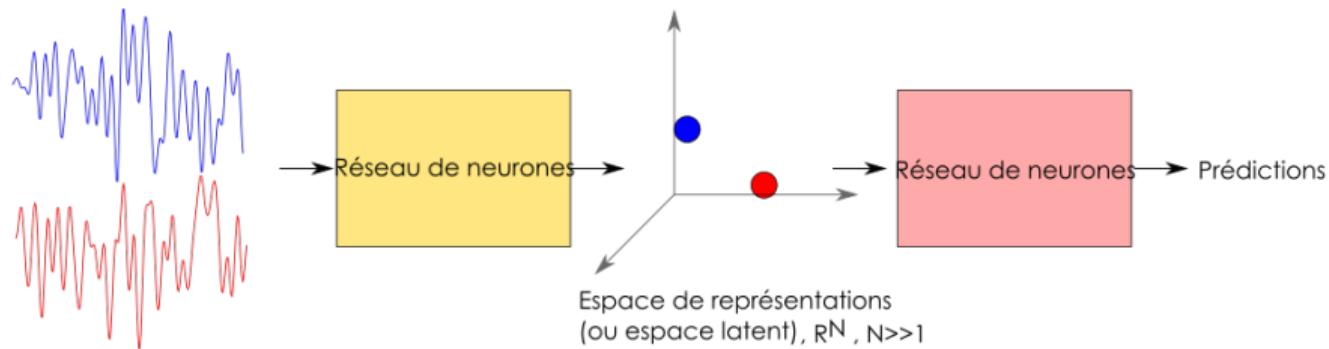
Espace Latent

Trois étapes:

- Extraction de features temporelles
- Contraction dimension temporelle
 - Exemple : Par moyenne
- Classification à proprement parler



Espace Latent



Apprentissage de métrique

Metric learning = ensemble de techniques destinées à concevoir un espace latent qui a de bonnes propriétés topologiques.

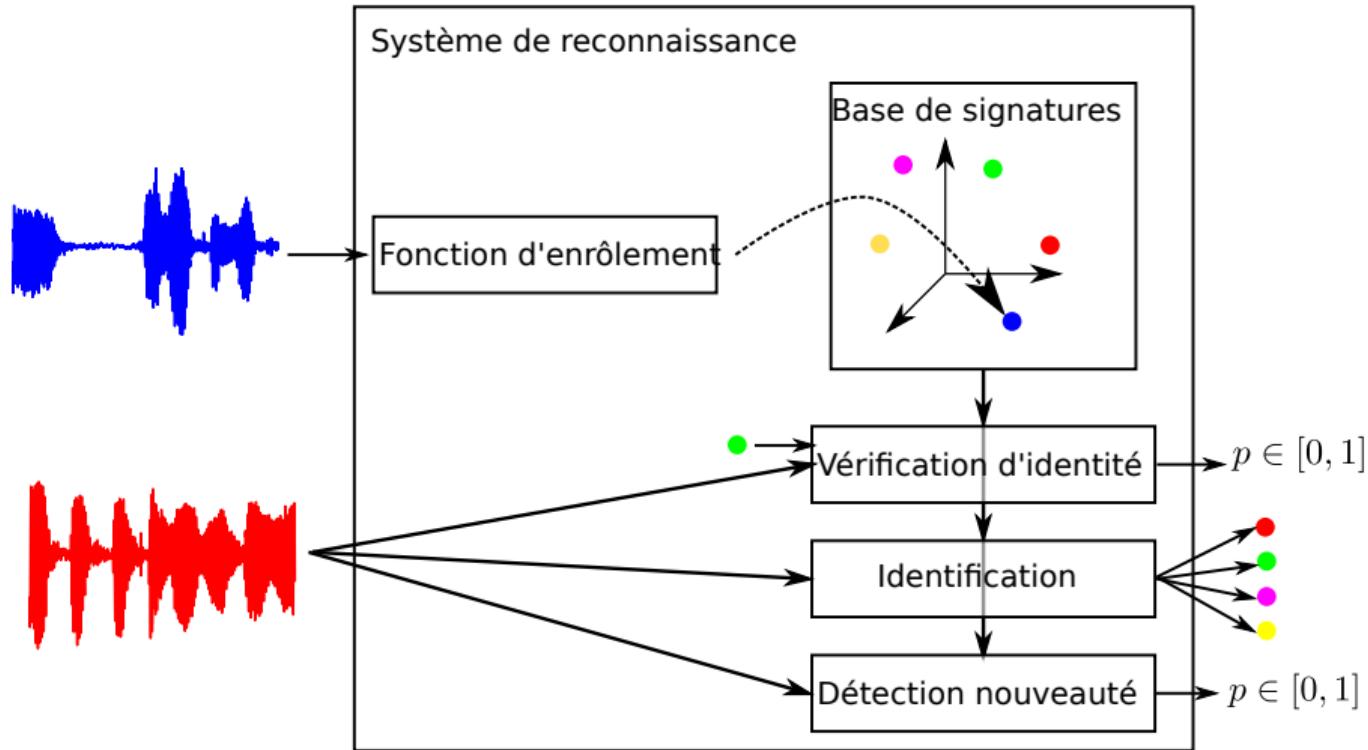
Si f réseau de neurones, x, y deux données ,

$$d_{\text{latent}}(f(x), f(y)) \simeq \frac{1}{\text{Similarité}(x, y)}$$

Permet :

- Interpoler un point entre deux données
- Ajouter des classes à un système de reconnaissance
- Déetecter des anomalies

Apprentissage de metrique



Apprentissage de métrique - Pairwise loss

Batch = paires de données $(x_i^{(0)}, x_i^{(1)})$, x_1 avec $\delta_i = 1$ si $x_i^{(0)}$ et $x_i^{(1)}$ proviennent de la même classe et 0 sinon.

$$\mathcal{L}_{\text{pairwise}} = \sum_i \delta_i \|f(x_i^{(0)}) - f(x_i^{(1)})\| + (1 - \delta_i) \max(m - \|f(x_i^{(0)}) - f(x_i^{(1)})\|, 0)$$

m = marge = hyperparamètre

Apprentissage de métrique - Triplet loss

Batch = de triplets de données $(x_i^{(a)}, x_i^{(p)}, x_i^{(n)})$, avec

- $x_i^{(a)}$ et $x_i^{(p)}$ dans la même classe et
- $x_i^{(a)}$ et $x_i^{(n)}$ dans des classes différentes

$$\mathcal{L}_{\text{triplet}} = \sum_i \max \left(||f(x_i^{(a)}) - f(x_i^{(p)})|| - ||f(x_i^{(a)}) - f(x_i^{(n)})|| + m, 0 \right)$$

m = marge = hyperparamètre

Prototypical Networks - inférence

Snell et Al., *Prototypical Networks for Few-shot Learning*, NIPS 2017

Routine d'apprentissage intégrant :

- l'apprentissage métrique
- la notion d'enrôlement d'une classe
- l'introduction de nouvelles classes à partir d'un nombre restreint de données (few shot learning)

3 modes d'utilisation :

- Enrôler une nouvelle classe
- Rattacher une nouvelle donnée à une classe précédemment enrôlée
- Apprentissage

Prototypical Networks - inférence

A l'inférence, on suppose qu'on a un réseau de neurones f qui plonge des données dans un espace latent.

- Je veux enrôler K classes C_k à partir de N_k données de références $x_i^{(k)}$:
 - $C_k \mapsto c_k = \frac{1}{N_k} \sum f(x_i^{(k)})$
- J'ai une nouvelle donnée x , je veux calculer la probabilité qu'elle appartienne à la classe k :
 - $p(C(x) = k|x) \propto \exp ||f(x) - c_k||$

Setting de l'apprentissage:

- K classes avec N données disponibles pour chaque.
- L : hyperparamètre, $L < K$ le nombre de classes vues par itération d'apprentissage.
- N_S le nombre de données support et
- N_Q le nombre de données de requête utilisées pour chaque classe, $N_S + N_Q < N$

Prototypical Networks - apprentissage

On initialise un réseau de neurones f_θ

- Pour toute itération:
 - Choisir L classes parmi les K disponibles: k_0, \dots, k_{L-1}
 - Pour chaque k_i :
 - Tirer un sous ensemble S_{k_i} de N_S éléments dans C_{k_i}
 - Tirer un sous ensemble Q_{k_i} de N_Q éléments dans $C_{k_i} - S$
 - Calculer $c_{k_i} = \frac{1}{N_S} \sum_{c \in S_{k_i}} f_\theta(c)$

On optimise alors la fonction de perte

$$\sum_i \sum_{c \in Q_k} \left(d(f(c) - c_{k_i}) + \log \sum_{j \neq i} \exp(-d(f(c), c_{k_j})) \right)$$

Séparation de sources

Séparation de sources - multicapteurs

M signaux sources : $x_i(t)$ qu'on observe sur N capteurs :

$$\begin{pmatrix} y_0(t) = \sum_{i=0}^{M-1} a_{i,0} x_i(t - \delta_{i,0}) \\ y_1(t) = \sum_{i=0}^{M-1} a_{i,1} x_i(t - \delta_{i,1}) \\ \vdots \\ y_{N-1}(t) = \sum_{i=0}^{M-1} a_{i,N-1} x_i(t - \delta_{i,(N-1)}) \end{pmatrix}$$

Exemples:

- Séparation de locuteurs
- Séparation voix - bruit

Séparation de sources monaurale

$$y(t) = x_0(t) + x_1(t)$$

avec y issu d'un seul capteurs

Problème sous-déterminé \Rightarrow besoin d'injecter des informations a priori pour tenter de le résoudre.

Exemple : débruitage, $y(t) = x(t) + b(t)$

- $x(t)$ signal utile (ex: voix)
- $b(t)$ signal perturbateur

Evaluation d'une reconstruction de parole

$y(t) = x(t) + b(t)$, x signal objectif, \hat{x} signal estimé.

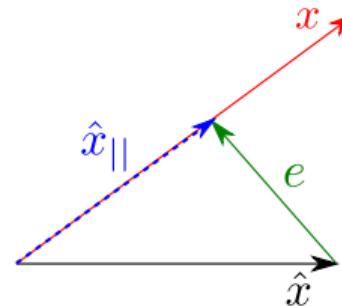
- SNR : Signal to Noise Ratio

$$\text{SNR} = 10 \log \frac{\|x\|^2}{\|\hat{x} - x\|^2}$$

- SI-SNR (Scale Invariant Signal to Noise Ratio)

$$\hat{x}_{||} = \frac{\langle \hat{x}, x \rangle x}{\|x\|} \quad e = \hat{x} - \hat{x}_{||}$$

$$\text{SI-SNR} = 10 \log_{10} \frac{\|\hat{x}_{||}\|^2}{\|e\|^2}$$



Evaluation de l'intelligibilité d'une reconstruction de parole

- Métriques subjectives:
 - MOS (Mean Opinion Score) calculé à partir d'un panel d'humains.
- Métriques objectives calculées informatiquement:
 - STOI (Short Time Objective Intelligibility) $\in [0, 1]$
 - PESQ (Perceptual Evaluation of Sound Quality) $\in [-0.5, 4.5]$
 - PESQ $\in [1, 2[$ impossible de comprendre
 - PESQ $\in [2, 2.4[$ effort considérable pour comprendre
 - PESQ $\in [2.4, 2.8[$ effort modéré pour comprendre
 - PESQ $\in [2.8, 3.3[$ attention nécessaire et effort léger
 - PESQ $\in [3.3, 3.8[$ attention nécessaire
 - PESQ $\in [3.8, 4.5[$ aucun effort requis

Débruitage - Filtre de Wiener

Approche classique de traitement du signal pour le débruitage:

$$y(t) = x(t) + b(t)$$

avec x et b indépendants et de DSP connues.

On cherche un filtre linéaire h tel que

- $\hat{x} = h * y$
- et $\mathcal{L} = \mathbb{E} (\int |\hat{x}(t) - x(t)|^2 dt)$ minimal

Débruitage - Filtre de Wiener

$$\mathcal{L} = \mathbb{E} \left(\int |(h * s)(t) + (h * b)(t) - s(t)|^2 dt \right)$$

$$\mathcal{L} = \mathbb{E} \left(\int |\text{TF}(h)\text{TF}(s)(f) + \text{TF}(h)\text{TF}(b)(f) - \text{TF}(s)(f)|^2 df \right)$$

$$\mathcal{L} = \int \mathbb{E} \left(|\text{TF}(h)\text{TF}(s)(f) + \text{TF}(h)\text{TF}(b)(f) - \text{TF}(s)(f)|^2 \right)$$

⇒ minmial si le terme intégré l'est pour tout f

$$\text{TF}(h)(f) = \frac{\text{DSP}(s)(f)}{\text{DSP}(s)(f) + \text{DSP}(b)(f)}$$

Débruitage - Filtre de Wiener

$$\text{TF}(h)(f) = \frac{\text{DSP}(s)(f)}{\text{DSP}(s)(f) + \text{DSP}(b)(f)}$$

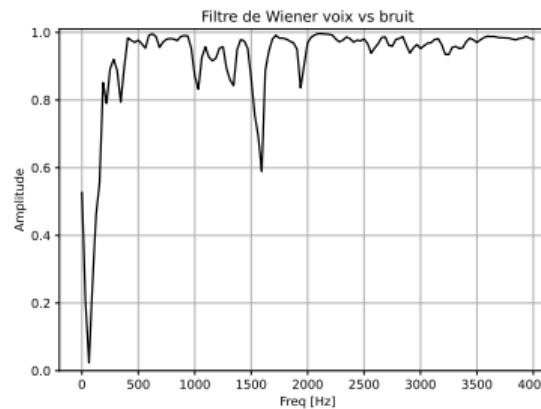
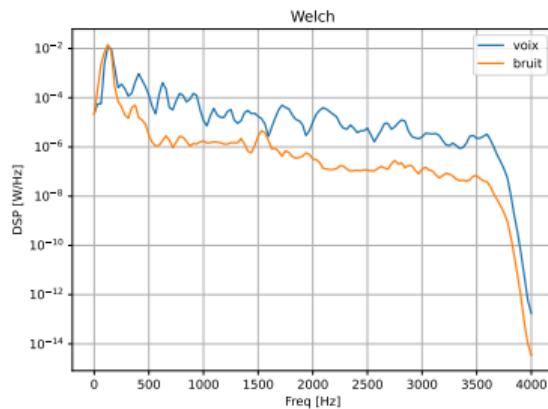
Si b bruit blanc, $\text{DSP}(b) = \sigma^2$

$$\begin{aligned}\text{TF}(h)(f) &= \frac{1}{1 + \frac{\sigma^2}{\text{DSP}(s)(f)}} \\ &\simeq 1 && \text{si } \text{DSP}(s)(f) \gg \sigma^2 \\ &\simeq 0 && \text{si } \text{DSP}(s)(f) \ll \sigma^2\end{aligned}$$

Débruitage - Filtre de Wiener

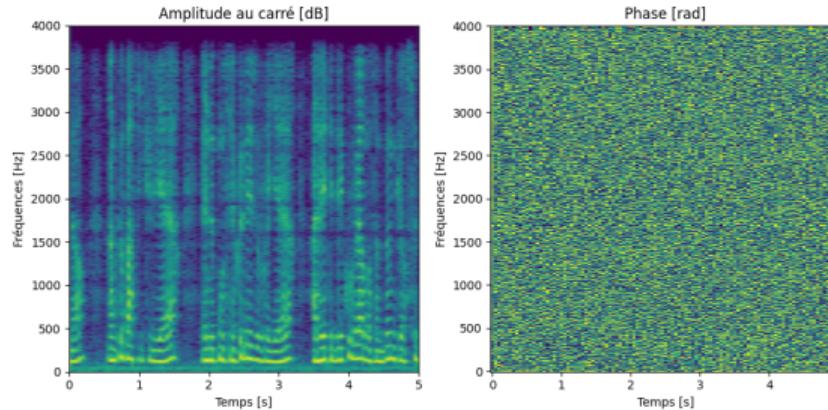
$$\text{TF}(h)(f) = \frac{\text{DSP}(s)(f)}{\text{DSP}(s)(f) + \text{DSP}(b)(f)}$$

Mélange = voix + bruit de moteur



Analyse - synthèse

TFCT = suite des transformées de F. sur des fenêtres de temps successives.



Signal $\xrightarrow{\text{TFCT}}$ Domaine temps-fréquence $\xrightarrow{\text{TFCT}^{-1}}$ Signal

```
from scipy.signal import stft , istft  
fe=...  
x=...  
nperseg = 512 # nombre d'échantillons dans la fenêtre  
nfft=nperseg # nombre de fréquences calculées par fenêtre  
nooverlap=0 # recouvrement entre deux fenêtres
```

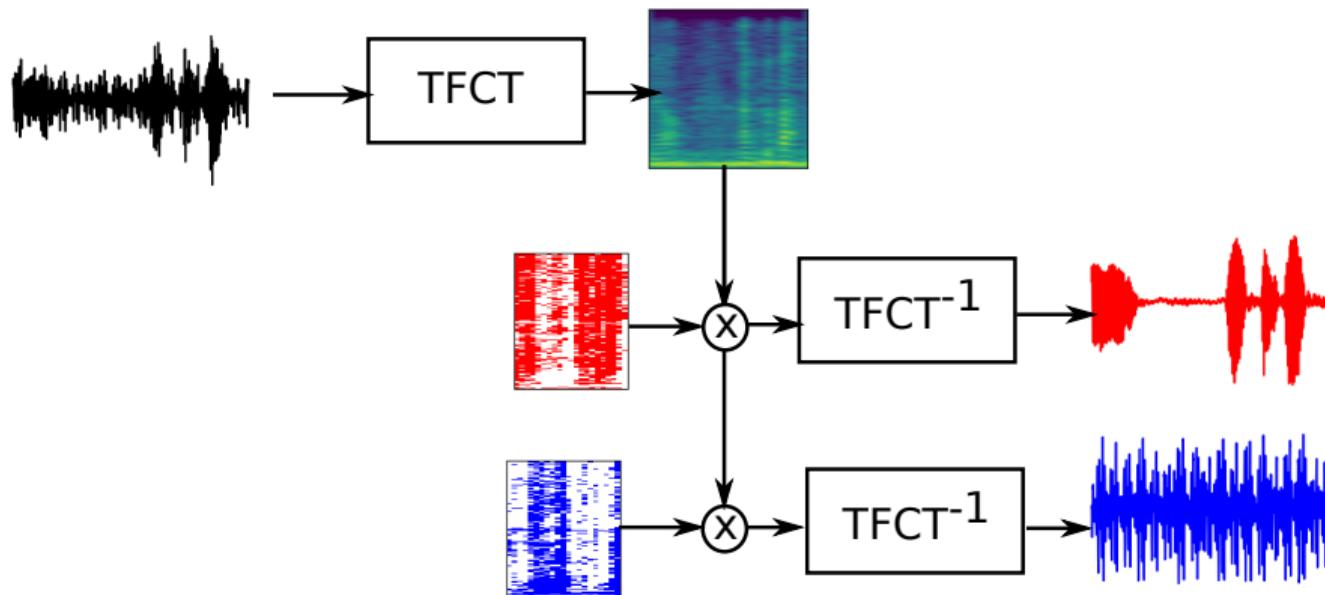
```
f,tfct = stft(x, fs=fe, nperseg=nperseg,nfft=nfft,nooverlap=nooverlap)  
t, x_est = istft(tfct fs=fe, nperseg=nperseg,nfft=nfft,nooverlap=nooverlap)  
#x_est = x
```

La TFCT peut-être paramétrée de façon *redondante*:

- nperseg = 400
 - nfft = 512
 - nooverlap = 100
- ⇒ La TFCT redondante est toujours inversible

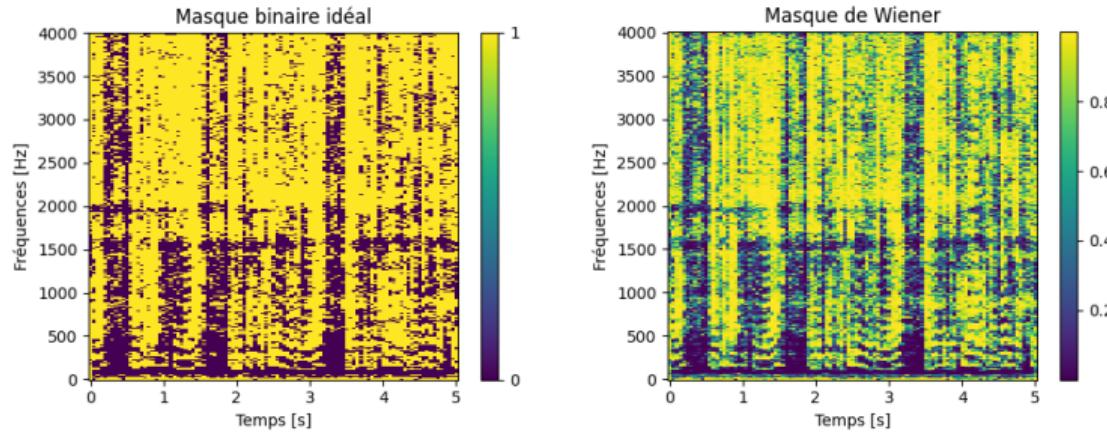
Séparation de sources monaurale -Approches par masquage

Idée estimer un masque temps-fréquence pour chaque source et l'appliquer à la TFCT du mélange pour reconstituer les sources



Séparation de sources monaurale - Masques orcales

Oracles = connus a priori par construction des données

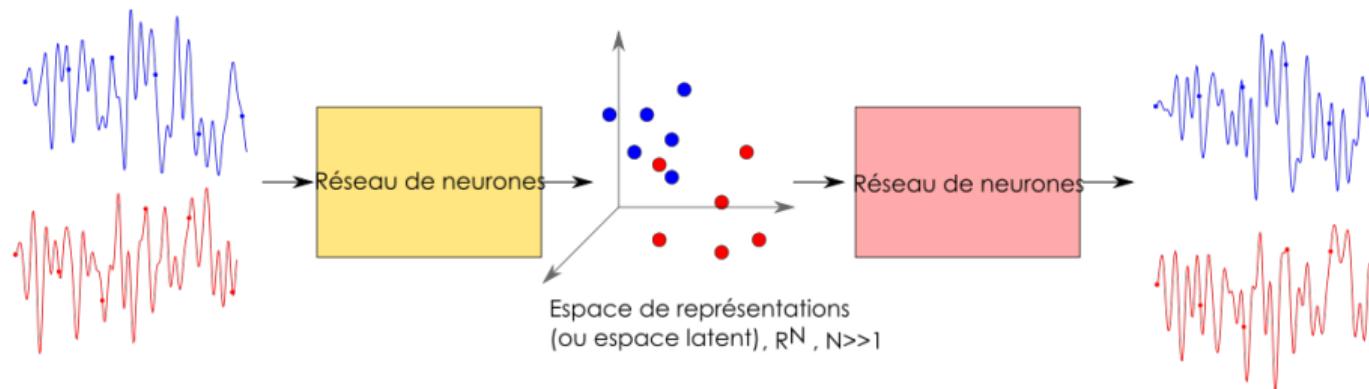


La question : comment en pratique estimer ces masques à partir du mélange ?

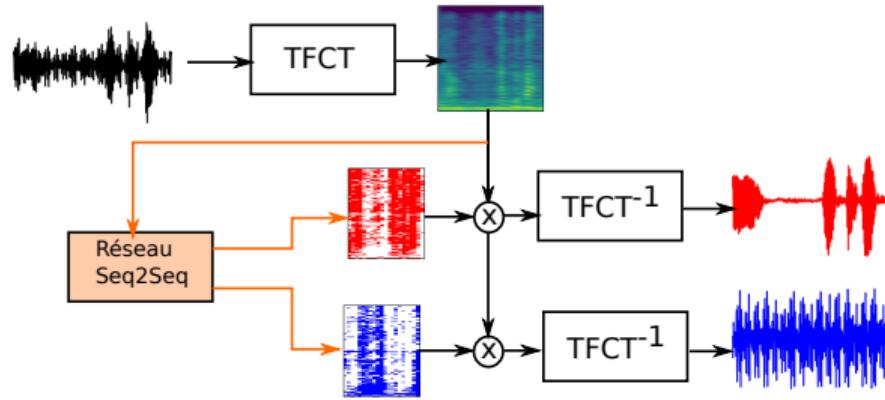
Réseaux Seq2Seq

Réseau F qui transforme un signal de n échantillons en un autre signal de n échantillons.

$$y_i = F(x_i|x)$$



Masking networks



On utilise un réseau de neurones F Seq2Seq pour estimer les masques $M_k(t_i, f_i)$.

- $M_k = 0, 1$: masques binaires
- $M_k \in [0, 1]$: masques pondérés (sigmoïde)
- $\sum M_k = 1$: masques normalisés (softmax)

Exemple: $F = \text{LSTM}$

Réseau F qui transforme un signal de n échantillons en un autre signal de n échantillons.

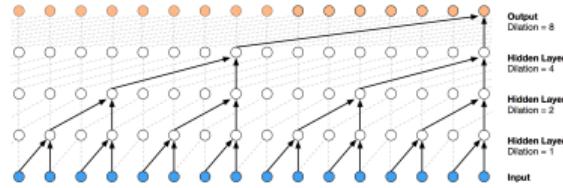
$$y_i = F(x_i|x)$$

Champs perceptif: Nombre d'échantillons de x dont dépend effectivement y_i

- Convolution noyau de taille $K = 2k + 1$, N couches: $\pm N \cdot k$
 - Réseau récurrent: mono-directionnel: tout le passé (en théorie, en pratique 50)
 - Réseau récurrent: bi-directionnel: tout le signal (en théorie, en pratique les ± 50)
- ⇒ Objectif : avoir un grand champs perceptif

Convolution dilatées

D'une couche à une autre, "dilate" le noyau des convolution (ie le bourre de 0)
Solution pour agrandir le champs perceptif sans décimer le signal



```
pointwise = torch.nn.Conv1d(in_channels=1, # entrée [B,1,T]
                           out_channels=4, # sortie [B,4,T]
                           kernel_size=11, # préférer les nombres impairs
                           dilation=2, # paramètre de dilatation
                           padding='same'
                           )
```

Couches d'Attention

Popularisé en NLP par [Vaswani et Al., Attention is all you need, NeurIPS 2017](#).

Idée de base: transformer (x_i) en (y_i) selon:

$$y_i = \sum_j \langle x_i, x_j \rangle x_j$$

On ajoute des paramètres pour gagner en expressivité

$q_i = Q(x_i)$ $k_i = K(x_i)$, $v_i = V(x_i)$ avec Q, K, V à apprendre.

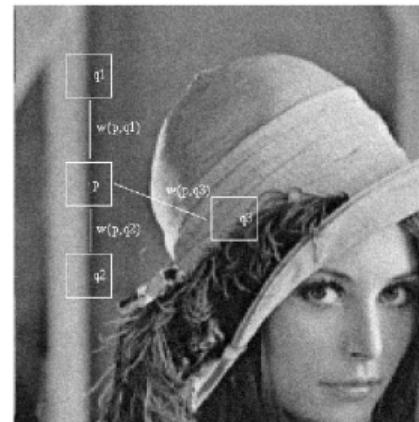
Puis on pose $P_i = \text{softmax}((\langle q_i, k_j \rangle)_j)$ et finalement

$$y_i = \sum_j P_{ij} v_j$$

Couches d'Attention

A la base des Transformers et des LLM actuels.

Mais similarités avec l'idée des *Non Local Means* en débruitage d'images



Buades A., Morel J.M., *A non-local algorithm for image denoising*, CVPR 2005 .

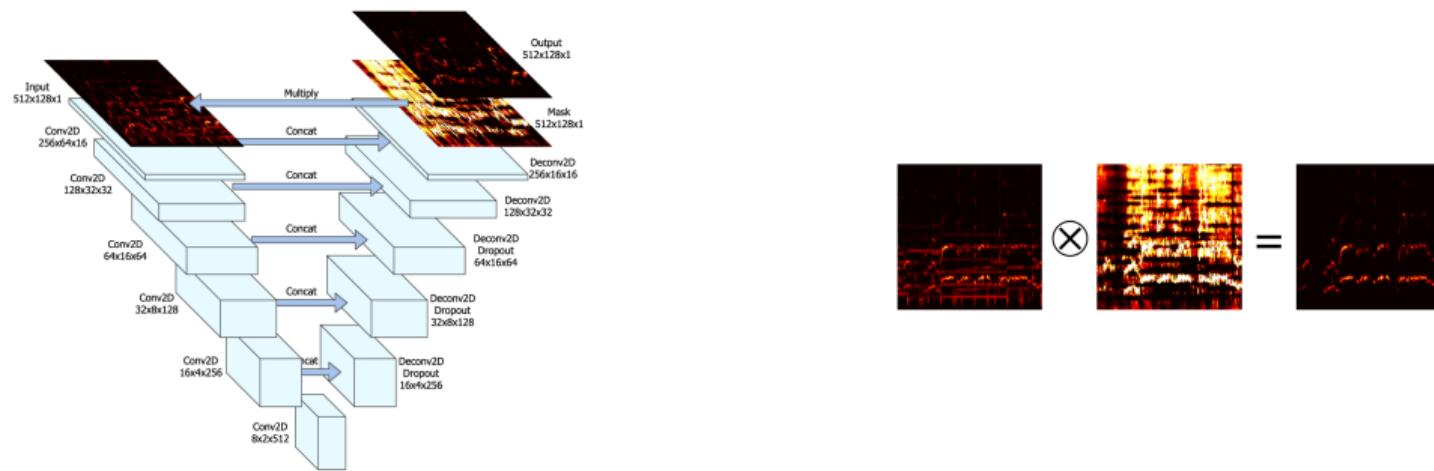
U-Net

Architecture convolutionnelle (2D) introduite en segmentation sémantique d'images.

Structure de type encodeur - décodeur avec des courts-circuits entre les deux.

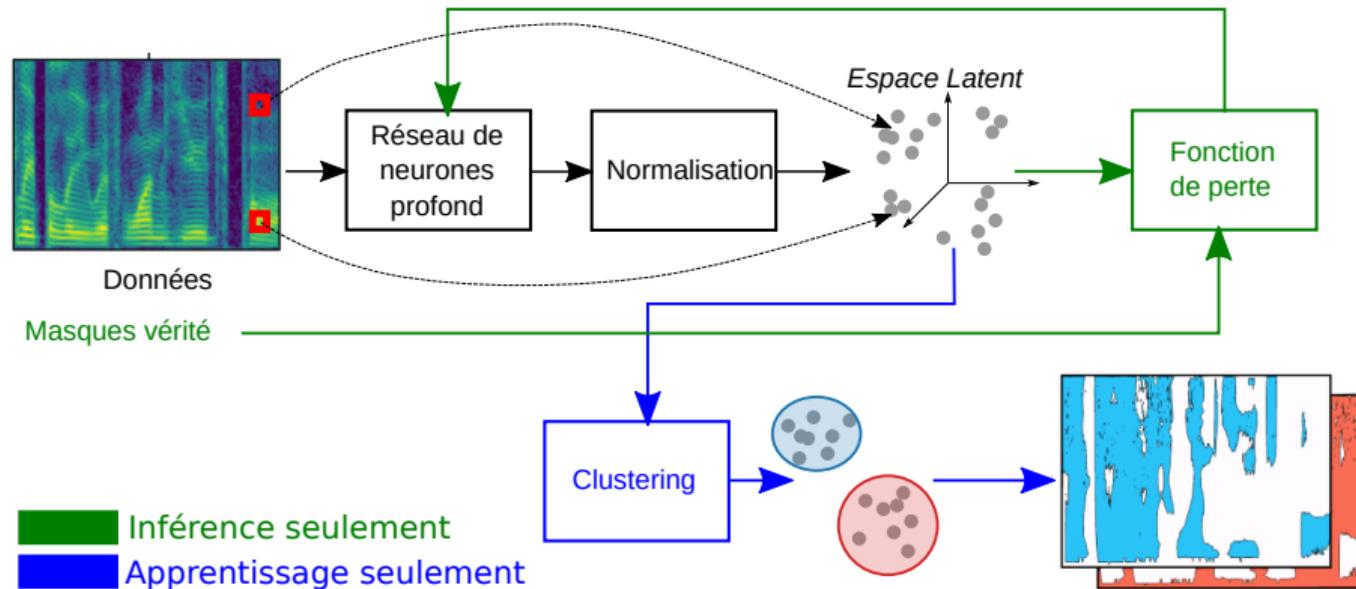
Ronneberger et Al., *U-net: Convolutional networks for biomedical image segmentation*, 2015

Janson et Al., *Singing voice separation with deep U-Net Convolutional Networks*, ISMIR 2017



Deep Clustering

Hershey, Chen, Le Roux, Watanabe, Deep clustering: Discriminative embeddings for segmentation and separation, ICASSP 2016



- Labels = masques idéaux: $Y_{ij} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ si le pixel temps-fréquence ij correspond à la source 0, $Y_{ij} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ si le pixel temps-fréquence ij correspond à la source 1
- $V_{ij} = F(X_{ij}|X)$: plongement dans l'espace latent d'un pixel temps-fréquence.

Deep Clustering

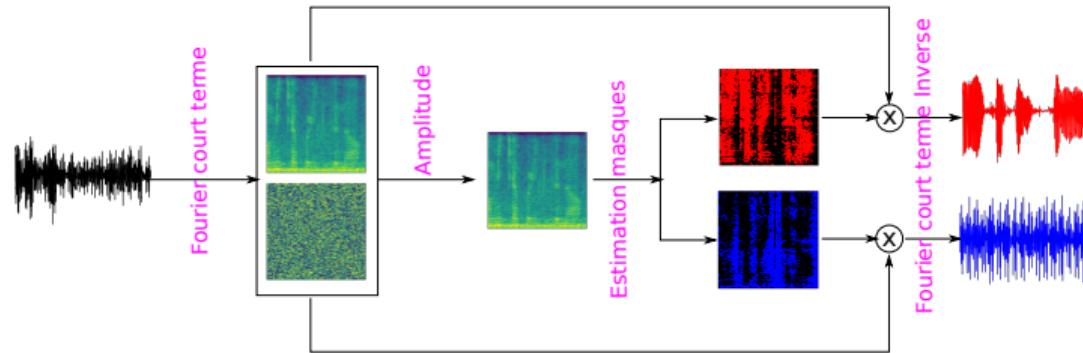
- $Y = (Y_{ij})$, $Y \cdot ({}^t Y)$ matrice vaut 1 en ij, kl si ij et kl appartiennent à la même source
- $V = (V_{ij})$, $V \cdot ({}^t V)$ matrice des produits scalaires entre paires V_{ij}, V_{kl}

Fonction de perte: Norme de Frobenius:

$$\mathcal{L}(V, Y) = |V \cdot ({}^t V) - Y \cdot ({}^t Y)|^2 = \sum_{ij, kl, Y_{ij} = Y_{kl}} (\langle V_{ij}, V_{kl} \rangle - 1)^2 + \sum_{ij, kl, Y_{ij} \neq Y_{kl}} \langle V_{ij}, V_{kl} \rangle^2$$

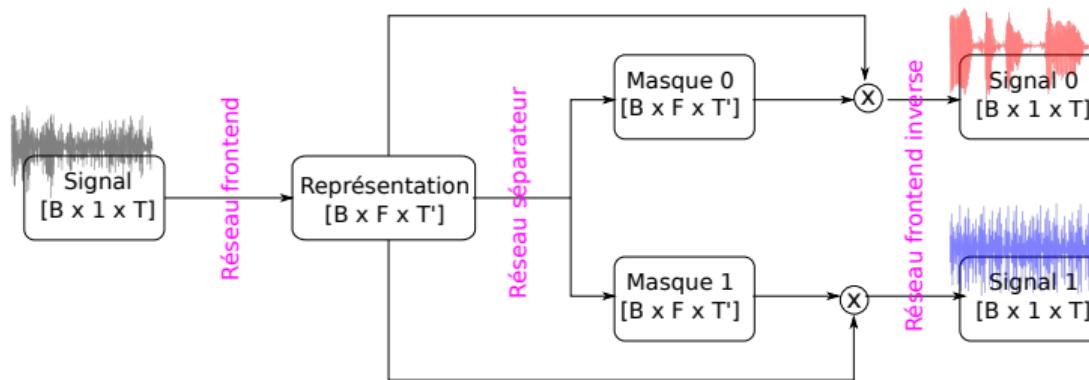
Luo et Al., *TaSNet: Time-Domain Audio Separation Network for Real-Time, Single-Channel Speech Separation*, ICASSP 2018

Luo et Al., *Conv-TasNet: Surpassing Ideal TimeFrequency Magnitude Masking for Speech Separation*, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2019



Luo et Al., *TaSNet: Time-Domain Audio Separation Network for Real-Time, Single-Channel Speech Separation*, ICASSP 2018

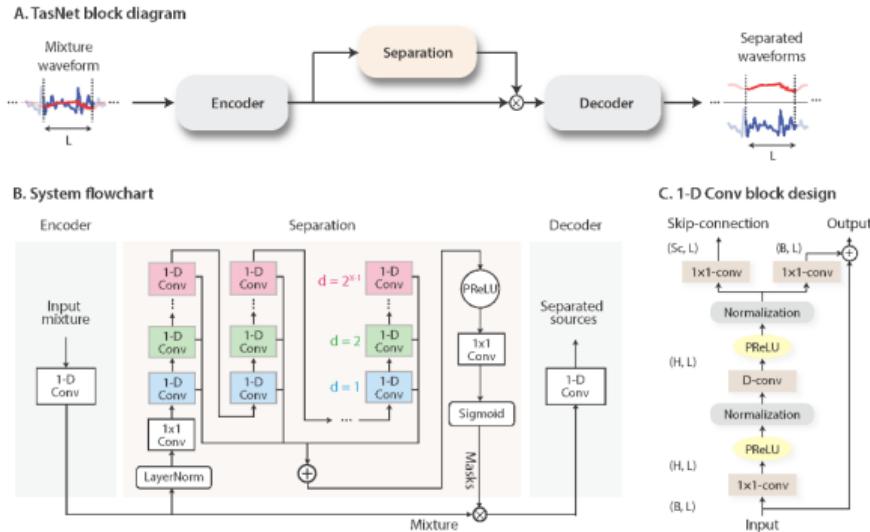
Luo et Al., *Conv-TasNet: Surpassing Ideal TimeFrequency Magnitude Masking for Speech Separation*, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2019



TasNet

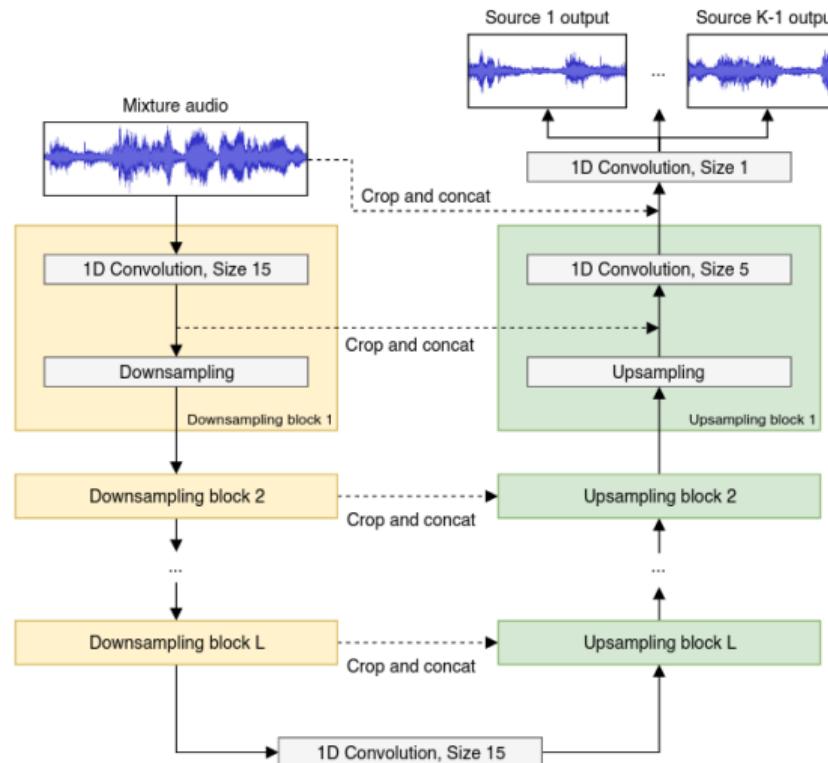
Luo et Al., *TasNet: Time-Domain Audio Separation Network for Real-Time, Single-Channel Speech Separation*, ICASSP 2018

Luo et Al., *Conv-TasNet: Surpassing Ideal TimeFrequency Magnitude Masking for Speech Separation*, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2019



Wave UNet

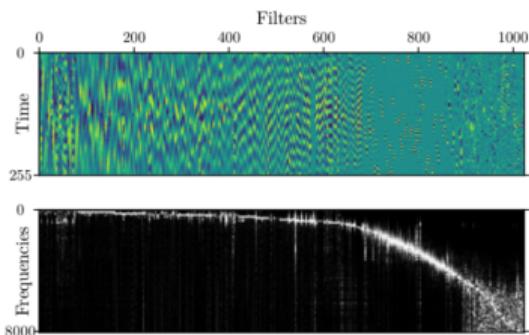
Stoller et Al., WAVE-U-NET: a multi-scale neural network for end-to-end audio source separation, ISMIR 2018



Physically informed networks pour la séparation de sources

Mathieu et Al., *Phase shifted Bedrosian filterbank: an interpretable audio front-end for time-domain audio source separation*, ICASSP 2022

- Gammatones : $\gamma(t) = at^{\textcolor{red}{n}-1} e^{-2\pi i \textcolor{red}{b}t} \cos(2\pi \textcolor{red}{f}t + \phi)$
- Filtres analytiques $\alpha(t) = \textcolor{red}{f}(t) + i\mathcal{H}(f)(t)$
 - \mathcal{H} transformée de Hilbert
- Hilbert étendue (généralisation du tiret précédent)
- Filtres de Bedrosian : $\zeta_k(t) = \textcolor{green}{A}(t) \cos(2\pi \textcolor{red}{f}t + \frac{k\pi}{K})$
 - K nombre de phases = hyperparamètre
 - On apprend plusieurs familles (ζ_k)
 - $\textcolor{green}{A}(t) = \textcolor{red}{a}(t) * \exp(-(\frac{t\textcolor{red}{f}}{\sigma}))$



Physically informed networks pour la séparation de sources

Mathieu et Al., *Phase shifted Bedrosian filterbank: an interpretable audio front-end for time-domain audio source separation*, ICASSP 2022

Domaine	Type	Fenêtre 128 ech	Fenêtre 1024 éch.
		SISNR	
Rééls	Filtres aléatoires	9.9	9.1
	Filtres libres	10.4	10.1
	Gammatones	10.4	6.5
Complexes	Filtres analytiques	10.3	10.7
	Hilbert étendu	10.4	10.7
Rééls	Bedrosian	10.5	10.8